

Forust

A 2009 Summer Project By Neil C. Obremski

Disclaimer

This document intends to provide a structure to the labor involved as well as an explanation of the project, its parts, and its stages. You might find aspects of the writing common sense or terribly tedious, because *it is written from me to me*. Here I lay the law to train my tenacious tendency to meander into a focused force of freethinking.

Introduction

Forust is a website fostering web page creation and development through simplistic input controls, strong URL permanence, and clean XHTML output.

Something more action-based?

Anyone can go to forust.com, type stuff in a box, upload some file, and click a button to publish it for everyone to see and append to.

Here are the three critical points which every piece will encourage and reinforce. I wrote them at the start of 2009:

1. Anonymous-Friendly
2. Comments are Content
3. Pictures with Captions

It is my hope to use it for favorably, and for the long-term, extending the world wide web.

Delivering a synopsis on a seemingly simple web development excursion is not as easy as it sounds, but giving a passerby overview is important if not horribly painful. When I originally came up with the idea, it hit me like a lightning bolt in the middle of the night.

Articulating it later made apparent its distressing similarity to existing services. Fuck it, I say, call it a "CMS" because it (and everything else) is in a way.

Rules

These aren't guidelines, *these are the law!* They apply for the duration of the schedule.

- 10 work weeks for approximately one minor version increase per week (0.0 to 1.0).
- 10 feature **stages** to parallel the week-based schedule.
- Technology: PHP 5 (server-side code), MySQL 5 (caching), S3 (permanent storage), MediaTemple (hosting), XHTML Strict (output)
- No stage longer than a week.
- Public-visible file extension: ".html"
- Design for Chrome (webkit), FireFox (gecko), and Kindle (limited/pda).
- Subversion for version control; use neilstuff repository on Google Code.
- Google Code for all issue tracking.
- Avoid other product comparisons in documentation; let people draw their own conclusions.

Technological cornerstones include ...

- URL permanence.
- Unicode everywhere.
- No CAPTCHA's.
- Web site *is* an API service.

Non-goals / future features ...

- Testing (ouch, but that's why I'm considering this an alpha).
- Theme, logo (+favico), and CSS sugar.
- Monetization of any kind.
- Upload proxy (versus direct-upload to S3).
- Translation (automated, or otherwise).
- Optimization, compression, etc. No clever text algorithms, gzipping, etc. It's possible I'll add caching because it'll be *necessary*.
- Legalese (privacy policy, terms of service, etc.)

No Internet Explorer

Internet Explorer is notoriously bad about XHTML; Microsoft is competing against itself, divided into three: IE6, IE7, and now IE8. Researching browser statistics gave me a nasty shock, IE6 is nowhere near gone. IE7 and IE6 are still head to head at the lead, and I have no reason to believe IE8 will change this significantly. FireFox has about 20% penetration which isn't great, but it's not bad.

I won't spend time meddling in IE's deficiencies, because they can be overcome past the finish line. For example, a special JavaScript could be included to manipulate the CSS and HTML for those users. It's my experience that they would tend to have JavaScript on, otherwise they're using another browser anyway (a la FireFox with NoScript).

Stage 0: Tools and Documentation

Fill out this document, obviously, and ...

- ~~Decide on IDE~~ [Gedit / TextPad]
 - [Komodo](#): Couldn't figure out how to really *get* it
 - [Mono Develop](#): More .NET oriented
 - [SCREAM](#): More about HTML than code
 - Decided on [Geany](#)
- ~~Coding Style / Conventions~~ [semi-PEAR, DocBlock]
 - PEAR ?
- ~~Create node in neilstuff SVN repository.~~
- ~~Install LAMP in Ubuntu on my Thinkpad.~~
- ~~Sketch layout and site map.~~
- ~~User Scenarios: who would use this and why~~
- ~~Task List: All the things done by visitor, author, moderator, and administrator~~

Here are things which need research:

- ~~Content MD5 on S3 form uploads.~~ S3 automatically creates an ETag that is an MD5 of the content, but it does not compute the MD5 itself.

...

Stage 1. Pages

Firstly we need to be able to create, add to, and view pages. A couple of details here: creating S3 direct-upload form, handling S3 upload-success-redirect to create the page object, base code structure for accepting a command (treating website *as* a service) and rendering output, downloading attachments from S3 and proxying them via a simple command (`foo.html?a=number/name`). Text is limited to whatever can fit in the "f0" meta tag and attachments can only be images up to 5mb in size.

Stage 2. Comments

What would content be these days without recourse to compliment or denounce it? Comments will be the first *leaf* feature, so this exercises not only another form of input and page data, but also underlying structure.

Stage 3. User State

How do you login and keep session data? My plan is to rely initially on either Gdata (Google) or Open ID and track with standard PHP sessions. Anonymous users, with cookies enabled, will get a long-lived session for temporary authorship. On the site-front this means a form for logging in/out and a "public terminal" / "remember me" checkbox.

Stage 4. Moderation

- Tool: Rename page (new URL; keep old URL).
- Tool: Add link to category limb and form to add a page to a category (on the page itself)
- Support "r" (redirect) HTTP header item.
- Support "o" (old URL) HTTP header item.
- Categorize: rename + add link
- RSS feed of new things required moderation.
- Lock a page.
- Support "l" (lock) HTTP header item.
- Delete page part (mod).
- Destroy (admin).
- Undelete.

Gotta put a stop to anon-power-users at this point with the introduction of moderator status, a tool for managing *who* is a moderator (for admins), and also *admins* themselves. Moderators are stored in a PHP-writable data file whereas Administrators are specified in the site configuration (a private PHP file). Tools available to moderators: flagging/hiding pages or comments (reason must be given, from an enum), categorizing new pages, and listing uncategorized pages. Administrators can do all that as well as *permanently remove* content

(but not the URL) give a page a new URL *anytime*. Note that neither have permission to *change* content, which brings us to ...

Stage 5. Authorship

- User Levels:
 - Administrator (site)
 - Author (page(s))
 - Moderator (section)
 - Anonymous (limited)
- Editing page parts, permanent delete, permanent forwarding.
- Revision saving (every previous text value is saved).
- Attachment *never* edited, only text.
- Deleting main page part promotes part just beneath or creates a new one to fill?
- Moderation of owned entries (comments on, etc.)
- Create response page
- Open ID support.
- Persistence of customization session/cookie.

If you wrote it, then it's yours and your responsibility. Even anonymous users (assuming they have cookies enabled) retain authorship of their articles for a time. This stage adds an "author" leaf to pages which gives that person the ability to control follow-up contributions, lock out SE's (so they can display it on their own site without competing), specify a forward link which is displayed but not automatically used, mark a page as public/private (latter is never categorized, not shown in unmoderated pages list), and remove/edit content (change text, re-upload attachment). Edits are not allowed after comments are made against the current segments, otherwise they lose their context.

Stage 6. Rating

- Customization settings in a cookie, no login required.
- "Soft" delete / burn (flags entry, but doesn't destroy it).
- Thumb-up / star / favorite / etc.
- ~~Add comments to a page *part*, loaded via AJAX (e.g. won't show up on Kindle).~~ Not doing this, because comments are content.
- Democratic moderation.

Sure there's a lot of content, but is it *good* content? I have a hard time with ratings, because I feel in general they are done poorly and end up skewing the *perception* of quality. Amazon is one of the few sites which does ratings "ok", but I just don't believe in star ratings. The idea here will be more about customization: you can up vote / star / favorite items you *really* like. Alternatively you can "delete" (down vote) ones you don't. The data will be collated later into something more useful, but in the interim it will help you ignore things you don't want to see/read. Also, I want the ability for visitors to suggest a page's category (especially on new pages).

Stage 7. Videos

You may recall from the first stage that only *images* are supported, but video is a huge reason for me to put this site together. In this stage I'll increase the upload limit, the supported formats, write the player scripts (WMP, QuickTime, Flash), and come up with a

way to specify *embedded* videos (a la YouTube).

Stage 8. Logging and Statistics

- Record every action ... do web logs (Apache / IIS) suffice?
- Human views versus page views.
- Most popular.

I'll have to pull out my rusty MySQL at this point to store every tiny action that occurs. The logs provide some information for general page views, but I'd like to track more navigation to determine "human views", popularity, and quality (do the visitors *scroll* through the entire page, etc.). This stage is very ethereal, and I might end up replacing it with something more concrete... it seemed like a good idea, but now I'm wondering if better site structure / design should take precedence.

Stage 9. Scheduled Tasks

- Admin page for showing CRON info.
- Repair parent/child relationship of orphaned pages.
- Renaming process.
- Auto-decision for uncategorized / moderated pages with a lot of votes on way.

Things are bound to get out of sync, so I'll need to write a CRON job for scanning the pile of objects: site map, list orphans, invalid objects, corrupt page object, find errors in web server logs, and generate report(s) for these things.

Stage 10. Anti-Spam

All this hippy shit is fine and dandy, but the world isn't as clean as that. At this final stage in initial development, I'll explore methods of turning the tide on bots, scammers, phishermen, etc. The primary idea behind this is to allow the flow, but not to give it as much credit or consideration. Out of site, out of mind!

It's perfectly acceptable in my mind for other sites to utilize the output, or even for automated intelligence to add pages, but not for misanthropic deeds such as spamming, phishing, and infecting other users. Primarily the defense will be about *going with the flow*. That is, the content is a river that always floods and damming it is a pointless battle without end. I intend, rather, to hide and ignore like my spam filter does for me with email. Let it come in, just don't give it priority. That said, there are proactive things to be done which will be never-ending, just like the filters on spam technology are ever-changing.

Stage Idea Pool

Trackbacks: Detect links between pages and page-parts to create trackback lists. Additionally, utilize HTTP Referrer for outside trackbacks.

Adult content invisibility unless age is verified or at least "agreed" on.

Locking / petrification. Marks a page or part of page for permanent "archival" and

prevents edits on it. Comments may be shutdown as well, undecided there, especially for inflammatory posts.

Text mark-up. Initially I won't support anything besides text, not even line breaks, in a single page part. This idea is to allow for BBCode, Wiki, HTML, markdown, etc.

Health monitoring. Time being taken to render a page. Transient issues or permanent ones, logged somewhere. An RSS feed of changes.

Niche site output, e.g. category assigned to only show up on certain domains.

Thumbnails of pages when being listed in sections.

More attachment types: videos, music, pdf, ...

Content conversion.

REST API.

Generate PRC (ebook) from page(s).

Re-ordering of page parts.

Turk-based moderation.

Export page as MIME document/digest.